Journal of Data Analysis and Critical Management, Volume 01, Issue 01, 2025

Evaluation Of 'Cache' Memory Performance Variation On Size Variation

Amit Kumar[1], R. S, Kumar[2]

[1]Research Scholar/Amrita Vishwa Vidyapeetham, Coimbatore, Tamil Nadu, India.

[2]Department of Information Technology/Amrita Vishwa Vidyapeetham, Coimbatore, Tamil Nadu, India.

Abstract

To mitigate the discrepancy between processor and main memory speeds, 'cache's are added to a system. A 'cache' is a quick, tiny memory that sits between the main memory and the processor. The "cache"s access time is matched by the processor's cycle time. Therefore, the "cache' memory' of system should be able to reply to a memory request in about 10ns if the processor is operating at a 100MHz speed. "cache' memory' is actually constructed on the 'processor chip' and divided into separate instruction and data 'cache's in today's high-performance single-chip CPUs. These 'cache's typically have a size of 8 KB, so that the CPU chip has 16 KB of 'cache' overall. An off-chip 'cache', commonly known as the second-level 'cache' or L2 'cache', is also a common feature of system designs.

Keywords: Spatial Locality, Temporal Locality, CPU, Intel 'cache', Data Buffer.

Journal of Data Analysis and Critical Management (2025); DOI: XXXX.XXXX

Introduction

Access times for modern main memory chips range from 60 to 70 ns. The memory access time can rise to 100 ns or more when one includes the time it takes for a memory request to go from the CPU to the system bus, followed by the memory controllers and decode logic. The cycle time of a CPU operating at 100MHz is 10ns. An "ADD instruction" that takes one of its 'operands' from the main memory may wait 100 ns for that operand and complete the addition in 10 ns if we assume that an addition may be completed in a single processor cycle. The memory access time would then be the primary determinant of the total time needed to finish a program; a processor speed increase would have minimal impact.

The size of the L2 'cache' can range from 128 KB to 4 MB. The first-level or primary 'cache' is the name given to the on-chip 'cache'. The speed of the second-level 'cache' can be a little slower than that of the main memory, but the first-level 'cache' must match the speed of the processor. A memory request made by the processor initially travels to the primary 'cache'. A ''cache'' hit occurs if the data item is located in this 'cache'. A 'cache' miss occurs and the memory request is sent to the L2 'cache' if the data item cannot be located in the primary 'cache'. An L2 'cache' hit occurs and the data is returned to the primary 'cache' if the data item is located in this 'cache'. The request is ultimately sent to main memory if the data cannot be located in the L2 'cache'.

The data item is transmitted back to the L2 'cache' and subsequently the primary 'cache' after the main memory has responded to the memory request. Because the primary 'cache' can typically handle the memory request, 'cache's function effectively. Measurements really reveal that the data 'cache' can reply to the data request 85% of the time and the instruction 'cache' will hold the requested instruction 90% of the time. As a result, little access is made to the L2 and main memory.

Two facets of program behavior are responsible for the primary 'cache's ability to manage so many memory requests:

Corresponding Author: Amit Kumar, Research Scholar/Amrita Vishwa Vidyapeetham, Coimbatore, Tamil Nadu, India., e-mail: email

How to cite this article: Kumar, A., Kumar, R.S. (2025). Evaluation Of 'Cache' Memory Performance Variation On Size Variation. Journal of Data Analysis and Critical Management, 01(1):10-12.

Source of support: Nil

Conflict of interest: None

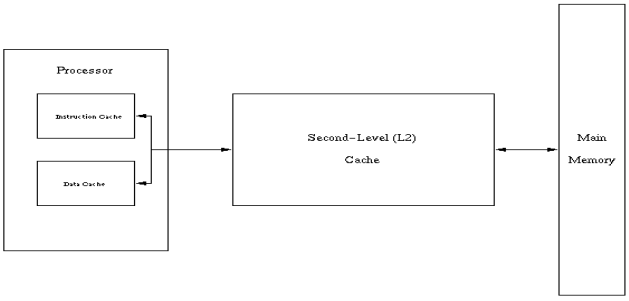

Figure 1: 'cache's in a typical system

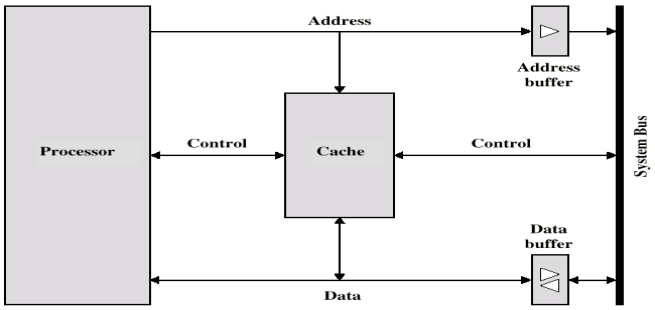

Figure 2: Typical 'cache' organization

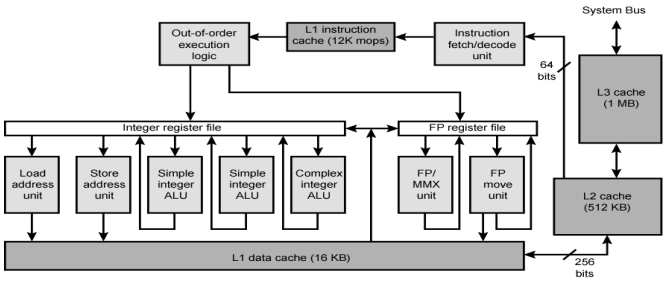

Figure 3: Pentium 4 Block Diagram

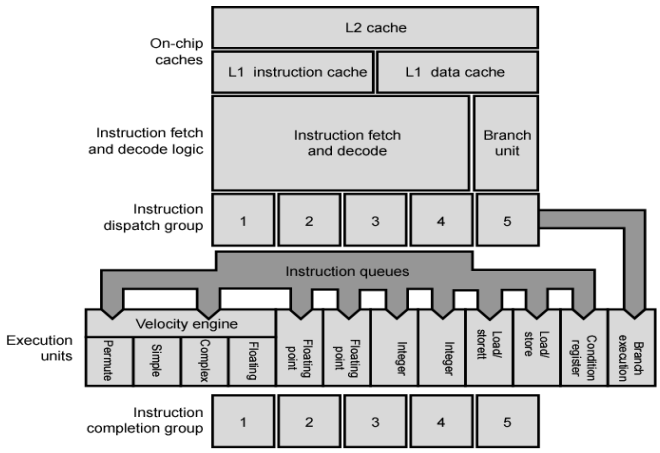

Figure 4: PowerPC G5 Block Diagram

Locality in Time

It is quite probable that a memory place will be mentioned again soon after it has been referenced.

Locality in Space

It is highly probable that a nearby memory location will be mentioned soon if a memory location is already referenced.

'cache' design parameter:

- 'cache' size;
- 'cache' block size;
- mapping function, which determines how blocks are assigned
- Write a Replacement Policy-algorithm for determining whether to replace blocks

Table 1: Intel 'cache' Evolution

Problem	*Solution*	*Processor on which feature first appears*
System bus speed is faster than external storage.	Utilize quicker memory technologies to add an external 'cache'.	386
The external bus becomes a bottleneck for 'cache' access as processor performance increases.	Processing at the identical speed as the processor, move the external 'cache' onto the chip.	486
Because of the chip's limited size, the internal 'cache' is quite modest.	Using quicker technology than main memory, add an external L2 'cache'.	486
When the Execution Unit and the Instruction Prefetcher both need access to the 'cache' at the same time, contention arises. Then, while the Execution Unit accesses the data, the Prefetcher is halted.	Make distinct 'cache's for instructions and data.	Pentium
The external bus becomes a bottleneck for L2 'cache' access as processor speed increases.	Make a distinct back-side bus that operates faster than the front-side external bus. The L2 'cache' is the focus of the BSB.	Pentium Pro
	Transfer the L2 'cache' to the CPU chip.	Pentium II
Certain applications work with enormous databases and require quick access to a lot of data. The 'cache's on the chip are too little.	Add external L3 'cache'.	Pentium III
	Move L3 'cache' on-chip.	Pentium IV

Table 2: Comparison of 'cache' Sizes

Processor	*Type*	*Year of Introduction*	*Primary 'cache' (L1)*	*2nd level 'cache' (L2)*	*3rd level 'cache' (L3)*
IBM 360/85	Mainframe	1968	16 to 32 KB	-	-
PDP-11/70	Minicomputer	1975	1 KB	-	-
VAX 11/780	Minicomputer	1978	16 KB	-	-
IBM 3033	Mainframe	1978	64 KB	-	-
IBM 3090	Mainframe	1985	128 to 256 KB	-	-
Intel 80486	PC	1989	8 KB	-	-
Pentium	PC	1993	8 KB	256 to 512 KB	-
PowerPC 601	PC	1993	32 KB	-	-
PowerPC 620	PC	1996	32 KB/32 KB	-	-
PowerPC G4	PC/server	1999	32 KB/32 KB	256 KB to 1 MB	2 MB
IBM S/390 G4	Mainframe	1997	32 KB	256 KB	2 MB
IBM S/390 G6	Mainframe	1999	256 KB	8 MB	-
Pentium 4	PC/server	2000	8 KB/8 KB	256 KB	-
IBM SP	High-end server/ supercomputer	2000	64 KB/32 KB	8 MB	-
CRAY MTAb	Supercomputer	2000	8 KB	2 MB	-
Itanium	PC/server	2001	16 KB/16 KB	96 KB	4 MB
SGI Origin 2001	High-end server	2002	32 KB/32 KB	4 MB	-
Itanium 2	PC/server	2003	32 KB	256 KB	6 MB
IBM POWER5	High-end server	2004	64 KB	1.9 MB	36 MB
CRAY XD-1	Supercomputer	2004	64 KB/64 KB	1 MB	-

Conclusion

In conclusion, the hardware may compare all 512 tag registers with the most important 28 bits of the physical address when a memory access is needed. 512 comparators are used in these simultaneous comparisons. We have a 'cache' hit if any of these comparators return a match. The correct byte is then chosen using the least significant four bits of the physical address once the line that generated the hit has been read out of the 'cache'.

Unfortunately, the enormous number of bits in the tag field and the requirement for numerous comparators make it challenging to construct a fully-associative 'cache'. Limiting the number of memory blocks that can be kept in each 'cache' line is one way to make the issue simpler. This might be accomplished by taking into account that the memory is composed of 8kb blocks, each of which starts at an address that is divisible by 8192. Any one of these 8kb blocks may have its first 16-byte block loaded into the 'cache''s first line, its second 16-byte block loaded into the 'cache''s second line, and so on. The number of the 16-byte block within the 8kb block is the same as the number of the 'cache' line, therefore in that scenario, we simply need to store which 8kb block of memory the bytes in a specific 'cache' line correspond to.

References

ACEVEDO, M F A probabdlstic study of two-level storage hierarchies M S Th, U of Texas, Austin, Tex, Dec 1972.

AVEN, O I, ET AL Some results on distribution-free analysis of pagmg algorithms IEEE Trans Comptrs C-25, 7 (July 1976), 737-745

BASKETT, F, AND RAFIl, A The A0 reversion model of program pagmg behavior Tech Rep #STAN-CS- 76-579, Dept Comptr Sci, Stanford U, Stanford, Cahf. Oct 1976

BELADY, L A A study of replacement algorithms for virtual storage computers IBM Syst J..5. 2 (1960), 78-101

BELL, J, CASASENT, D, AND BELL, C G An mvestlgaUon of alternative 'cache' organizations 1EEE Tram Comptrs. C-23, 4 (April 1974), 346-35 !

BURVILLE, P J, AND KINGMAN, J F C On a model for storage and search J Appl Probablhty 10 (1973I, 697-70 !

COFFMAN, E G, AND DENNING, P J Operating System Theory Prentice-Hall, Englewood Chffs, N J, 1973